# Correlated 15 Hz Data

*Historical perspectives*

Mon, Nov 8, 2004

Front ends used in Fermilab Linac and other projects have always supported access to 15 Hz correlated data. This refers to the ability for a client to receive data collected at rates up to 15 Hz in synchronism with the 15 Hz Fermilab accelerator, preserving the ability to associate, or correlate, the data across multiple front ends. This note relates the history and the means of providing such support.

***Historical review***

The front end software support for correlated data originated with Fermilab controls even before there were distributed front ends here. In 1969 the support for data collection resided in the single computer that served both client and server sides of Linac accelerator controls. The SDS Sigma 2 computer was outfitted with a 3 MB Rapid Access Disk (RAD). (!) Following a 15 Hz interrupt, this computer read the data from hardware whose tentacles reached throughout the 500-foot-long Linac to bring in the signals, most of which were captured by sample-and-hold circuitry. The accelerator operated at 15 Hz, delivering a pulse of beam at a 0.06% duty cycle. The RF systems that accelerated the beam delivered power to the RF cavities in 400 $\mu$s pulses. The quadrupole power supplies delivered their focusing current in pulses lasting only 1 ms whose flattops were about 100 $\mu$s. All these elements were synchronized to provide the ability to deliver 15 Hz Linac beam pulses continuously. The control system was to read this data at the correct times to make sense out of the accelerator operation. Except for the use of a single controls computer, these elements still exist today.

Within the same Linac controls computer, one could invoke a single application page in the MCR that displayed a sampling of the data in a 15 Hz data pool of all Linac device readings. An early application page that was popular was called the Custom page, since it permitted the user to build a list of devices, one per display line, with associated readings and support for settings either by "keyboard interrupt" or by knob control. Many such pages could be built to support familiar access to sets of devices for tuning. There were other application programs written, such as status pages, but it was this Custom page that ultimately became the Parameter Page that is well known in Fermilab accelerator operations today.

All the data was read at 15 Hz. Today we might call it "all the data, all the time." And the 15 Hz was the accelerator's same 15 Hz, which was always designed to be synchronized to the power line 60 Hz in order to avoid confusing any power supply ripple with the real data. What mattered most were the readings of the accelerator components during the brief time the Linac beam passed through some 300 drift tubes on the 1.5 $\mu$s journey to attaining an energy of 200 Mev. To that end, sample-and-hold hardware served to capture the data at a time that was usually set near the middle of the Linac beam pulse. The controls computer received an interrupt timed 1 ms after Linac beam time, when the high power components of the Linac were quiet and the sample-and-hold outputs were still fresh.

The control system computer needed to monitor the health of the Linac, since it would be impossible for operators to monitor everything continuously. To that end, alarm logic was added to the controls computer software, so that selected devices could be checked for being within selected limits, and digital data could be checked for being in the expected nominal states. But there was too much data to monitor all devices at 15 Hz, so the job was throttled; only 3 ms was allotted for this job every cycle. In practice, this meant that all devices of interest could be monitored for alarms only at 3 Hz, still adequate for human interaction time although not the best for accelerator reaction time.

In time, certain types of accelerator tuning and trip recovery became missions for which the controls computer could provide assistance. An ability for performing closed loops came about to handle these jobs, relieving operators of many such tasks.

When the first distributed Linac control system was installed in late 1982, this same support was still needed. But the client support was now handled by Vax computers operating in communication with PDP-11 "front ends." For Linac, the PDP-11 front end would not be connected directly to the raw signals but would be connected to a network of some seventeen 68K-based microprocessors housed in Multibus I crates. The communication path used for connecting the distributed systems together was the master-slave SDLC protocol, in which a Primary node orchestrated selected data collection from the Secondary distributed systems that actually connected to the hardware interfaces. The Primary node used ethernet, a relatively new networking protocol at the time, to connect to the PDP-11 computer.

***Little consoles***

As part of the plan for a distributed system, local consoles were supplied at a number of physical locations along the 500 feet long Linac. Support was added for page applications, including the parameter page, to be called up on these small consoles, each of which includes a 16 line by 32 black-and-white alphanumeric display, a keyboard, and a suite of buttons and lights. The devices that can be placed on each display line can be local devices, or they could be from anywhere else in the Linac. Every device has a name that can be looked up with the aid of the network; broadcast the name request, and the node that has that name replies with its own node and channel number. The source code of the page application does not have to care whether the data of interest is local or not; the underlying system code accepts requests made by the page application and handles network commmunication details. When the page application subsequently calls for the data, the fresh data values collected on the present cycle, no matter where they came from, are returned. Interestingly, it was this support for data from other nodes that provided the basis for the later implementation of server node support (see below) for data requests and settings.

***Acnet data requests***

Data requests (in the Acnet RETDAT protocol) that arrived at the PDP-11 were passed on to the Primary node in something like what is now known as the Classic protocol. The Primary node broadcast the requests to all the Secondary nodes, each of which noted what it should furnish in response to the requests whenever the Primary node subsequently called for its reply contributions, which it did at rates up to 15 Hz, at a time in the cycle when the Secondary nodes had collected all their data and were ready to reply. All this preserved the support for correlated data across multiple front ends. Each Secondary node connected to a portion of the Linac hardware interfaces, and each node supported alarm scanning for its own devices at the full 15 Hz rate, including operation of a control line that can inhibit further Linac beam pulses when certain devices were found to be out of limits or tripped. It made no sense to continue to accelerate bad beam or needlessly irradiate accelerator components. Similarly, the closed loop logic was implemented in each Secondary node.

***Linac controls upgrade***

About ten years later, the Linac distributed control system was upgraded to use VME hardware, and Smart Rack Monitors (SRMs) were installed to actually interface to the data. Communication with the SRMs from each front end was made via the arcnet network, to which it was easy to interface. Communication with the Vax computers was made directly, by providing RETDAT protocol support in each front end. In order to increase the chance for the Vax computers to collect correlated data across multiple front ends, server node support was added, in which the Acnet requests were made from the consoles to one "server node," even though the actual signals may reside in other nodes. The server node communicated (via forwarding the RETDAT request via multicast) with whatever front ends were needed in order to collect all the responses from the contributing nodes together into a composite reply for delivery to the Acnet console. In this way, the requested data could be amassed into one reply, making it possible that an application could see it as correlated; *i.e.*, it was measured on the same beam pulse.

***15 Hz data pool***

Within each front end, all data is measured at 15 Hz, and all alarm scanning of this data is done at 15 Hz, with all activity initiated every 15 Hz cycle by the accelerator timing system. During the time the data is collected into the data pool, no requests are processed and no replies are delivered; it is impossible for an outside user to see a partially updated data pool. By updating all the data pool at once, more efficient processing can be used. One thinks of digitizing all 64 channels of a digitizer, not merely one channel here and there as needed by requesters. Updating the data pool does not depend upon the needs of requesters. All device data within the data pool is always seen as correlated data; all was measured on the same 15 Hz cycle. Fulfillment of requests is easy, because all the latest readings are present in memory at one time. Alarm scanning is also easy, because all the required data is available in memory for immediate access.

***Booster BLMs***

After the Linac controls had been used for support of other Fermilab controls needs, there came a time when Booster Loss Monitors were being upgraded, and it was important to acquire correlated data across 8 different BLM front ends, so that the loss monitor data of some 70 BLMs could be studied together, since they had all measured losses coming from the same accelerated beam. In addition, these data had to be collected at rates up to 15 Hz, since that is the maximum rate of the Booster accelerator.

Providing for correlated support here turned out to be a problem, because the Vax computers do not support accelerator-synchronous 15 Hz invocation of application pages. Each Vax console uses internally-sequenced application executions at delays of 60 ms, 70 ms, and 70 ms, yielding an average rate of 15 Hz, but since it is asynchronous with the accelerator timing system, two replies from a 15 Hz request can arrive at the Vax from the server node between two successive executions of the application that may be separated by 70 ms. Data replies received by the Vax are kept in a data pool; they are not queued. Once reply data has updated the Vax data pool, the previous data is lost. (It should be noted here that the future Java-based Acnet controls use queuing, in the sense that a call-back is made to an application when fresh data arrives from a front end.)

In order to deal with the need for correlated data in this client environment, new support was added to the front ends, making it possible for data to be delivered by the network at 7.5 Hz that includes two sets of data that have just been collected by the front end at 15 Hz. The application looks for new data at every “15 Hz” execution, only finding new data present in the data pool at about every other time. But the data from each front end had to be correlated, so a key was needed to assist in that correlation. To that end, the format of the reply data for such 7.5 Hz requests includes a 4-byte header, in which the first 2 bytes are a 15 Hz cycle number, and the second 2 bytes is a count of the number of sets of 15 Hz data that follows. This count, except for the first prompt reply to such requests, is always 2.

The 15 Hz cycle number was derived from the low 16 bits of the cycle number included in the ethernet message that is multicast at 15 Hz that includes the Tevatron clock events that have occurred within the last 15 Hz cycle. (This was originally needed for front ends that did not have Tevatron clock decoding built in.) This ethernet message is sent from a special front end at the time of the `0x0F` clock event, which occurs about 16 ms before Booster reset time, which in turn occurs about 2 ms before Linac beam. The interpretation of the cycle number for this purpose is that it announces the cycle number that is to be used as a "time stamp" key for the data reply header for the data measured on the cycle *following* the one during which the message is received. This means that one has about 16 ms of warning before the key applies. Note that it was unnecessary for all front ends to monitor this ethernet message at 15 Hz. One node receives it, then occasionally, say every 256 cycles, multicasts it to the other nodes that need it. In between, each node simply adds one to its internal copy of this key.

As a simple example of this time stamped scheme for providing correlated data across multiple front ends, consider a request for a single analog reading that is to be reliably collected by an application at 15 Hz. For such a request, the application must ask for 8 bytes of data: 4 bytes for the reply header and 4 bytes for the readings sampled on two successive cycles. The time stamp key in the first word applies to the first data value; that number plus one applies to the second value. Again, the first immediate reply for such a request will have a count of 1, as only one data value is meaningful; subsequent replies will have a count of 2. As an example of this simple example, try requesting 8 bytes at 7.5 Hz for the reading property of device `B:BREVNT`. This will allow a user to know the applicable Booster reset event number relating to each time stamp key value.

Note that in order to correlate data across multiple front ends using this technique, all data of interest must include a time stamp key in its reply data. All IRMs and all Linac PowerPC front end nodes support this scheme.

Additional documentation about the support for Booster BLMs, including the time stamp key scheme described above is available in documents on the web, in *Local Application BLMS*.